

LARNet: Latent Action Representation for Human Action Synthesis (Supplementary Materials)

Naman Biyani¹
 namanb@iitk.ac.in

Aayush J Rana²
 aayushjr@knights.ucf.edu

Shruti Vyas²
 shruti@crcv.ucf.edu

Yogesh S Rawat²
 yogesh@crcv.ucf.edu

¹ IIT Kanpur
 Kanpur, India

² Center for Research in Computer Vision
 University of Central Florida,
 Florida, USA

1 Network details

In this document, we provide the architecture details for all the components of the proposed network. It includes the content encoder (Table 2), motion encoder (Table 1), motion discriminator (Table 3), the video discriminator (Table 4), the motion generator (Table 5), motion integrator (Table 6) and the video decoder (Table 7).

We also show more qualitative results on various datasets and on various component's effect in the proposed approach. Apart from this, the supplementary material also contains a demo video for our Latent Action Representation for Video Prediction based videos.

2 Metrics

To measure our generated video's overall quality, we use *SSIM* [1], *PSNR* [2], *FVD* [3], [4] and *FID* [5]. We follow [5] for FVD computation and use a pre-trained Inception3D network to generate the embeddings. We follow [3] for FID computation, using the pre-trained InceptionV3 network on each frame of the video and averaging over the generated frames to get mean score of each video.

3 Synthetic Dataset

We create a synthetic dataset for validating our approach in a controlled setting. The synthetic dataset consists up to 2 objects per video. Each video has a unique constant Gaussian noise as background, which is fixed throughout the video. There are 6 total action classes

Name	Layer	Input	Kernel Dims (T×H×W)	Strides (T×H×W)	Output Dims (T×H×W×C)
AE-Conv1 (skip)	3D Conv	\hat{v}	$7 \times 7 \times 7$	$2 \times 2 \times 2$	$8 \times 56 \times 56 \times 64$
AE-MaxPool1	3D Max Pool	AE-Conv1	$1 \times 3 \times 3$	$1 \times 2 \times 2$	$8 \times 28 \times 28 \times 64$
AE-Conv2	3D Conv	AE-MaxPool1	$1 \times 1 \times 1$	$1 \times 1 \times 1$	$8 \times 28 \times 28 \times 64$
AE-Conv3 (skip)	3D Conv	AE-Conv2	$3 \times 3 \times 3$	$1 \times 1 \times 1$	$8 \times 28 \times 28 \times 192$
AE-MaxPool2	3D Max Pool	AE-Conv3	$1 \times 3 \times 3$	$1 \times 2 \times 2$	$4 \times 14 \times 14 \times 192$
AE-IncModule1	Inception Module	AE-MaxPool2	-	-	$4 \times 14 \times 14 \times 256$
AE-IncModule2	Inception Module	AE-IncModule1	-	-	$4 \times 14 \times 14 \times 480$
AE-MaxPool3	3D Max Pool	AE-IncModule2	$3 \times 3 \times 3$	$1 \times 1 \times 1$	$4 \times 14 \times 14 \times 480$
AE-IncModule3	Inception Module	AE-MaxPool3	-	-	$4 \times 14 \times 14 \times 592$
AE-IncModule4	Inception Module	AE-IncModule3	-	-	$4 \times 14 \times 14 \times 512$
AE-IncModule5	Inception Module	AE-IncModule4	-	-	$4 \times 14 \times 14 \times 512$
AE-IncModule6	Inception Module	AE-IncModule5	-	-	$4 \times 14 \times 14 \times 528$
AE-IncModule7	Inception Module	AE-IncModule6	-	-	$4 \times 14 \times 14 \times 832$
AE-IncModule8	Inception Module	AE-IncModule7	-	-	$4 \times 14 \times 14 \times 832$
AE-IncModule9	Inception Module	AE-IncModule8	-	-	$4 \times 14 \times 14 \times 1024$
AE-Conv4	3D Conv	AE-IncModule9	$3 \times 3 \times 3$	$1 \times 1 \times 1$	$4 \times 14 \times 14 \times 256$
AE-Conv5a	3D Conv	AE-Conv1	$3 \times 3 \times 3$	$1 \times 1 \times 1$	$8 \times 14 \times 14 \times 128$
AE-ReLU5a	ReLU	AE-Conv5a	-	-	$8 \times 14 \times 14 \times 128$
AE-Conv5b	3D Conv	AE-Conv3	$3 \times 3 \times 3$	$1 \times 1 \times 1$	$8 \times 28 \times 28 \times 128$
AE-ReLU5b	ReLU	AE-Conv5b	-	-	$8 \times 28 \times 28 \times 128$

Table 1: Network details for the motion encoder E_v , which is used to encode the input video \hat{v} into motion features \hat{e}_m . AE refers to the ActionEncoder. Note that the input column contains either the tensor used as input to the particular layer or the layer whose output is used as input. Also, note that the Inception module is that from [14]. Since the proposed method involves hierarchical transformation, there are two outputs from this network: ActionEnc-ReLU5a, ActionEnc-ReLU5b. The bottom section of the table indicates three extra convolutional layers, one for each output, used to force a uniform number of channels for all two encodings.

(1: move left to right, 2: move right to left, 3: move up to down, 4: move down to up, 5: split and move apart, 6: merge together). Action class 1-4 has one object per video while class 5 and 6 has two objects per video. The objects have random shape of triangle, circle or rectangle with equal probability. Triangle and rectangle object also have random rotation. The objects can have random size between 20-40 pixels and can randomly scale up or down through the video by a factor of 2x. There is an equal probability for the object to have constant or variable speed through the video. We generate 500 videos per class to get a total of 3000 unique videos. We use 420 videos per class in training and 80 videos per class in testing set.

We evaluate the ability of the proposed approach to model the motion dynamics on a synthetic dataset. Figure 4 shows some sample videos predicted using LARNet. We observe that the generated videos have a clearly visible motion across frames which represents the target class. The action generation has a stochastic nature and a variation in motion is also evident in the generated videos. This variation includes stochastic path of the generated motion and a change (zoom-in/zoom-out) in size of the object. LARNet is also able to capture the variations in the object shapes as it generates successive future frames while synthesizing the target motion.

Name	Layer	Input	Kernel Dims (T×H×W)	Strides (T×H×W)	Output Dims (T×H×W×C)
IE-Conv1a	2D Conv + Batchnorm	P^J	3×3	1×1	$112 \times 112 \times 64$
IE-ReLU1a	ReLU	IE-Conv1a	-	-	$112 \times 112 \times 64$
IE-Conv1b	2D Conv + Batchnorm	IE-ReLU1a	3×3	1×1	$112 \times 112 \times 64$
IE-ReLU1b	ReLU	IE-Conv1b	-	-	$112 \times 112 \times 64$
IE-MaxPool1	2D Max Pool	IE-ReLU1b	2×2	2×2	$56 \times 56 \times 64$
IE-Conv2a	2D Conv	IE-MaxPool1	3×3	1×1	$56 \times 56 \times 128$
IE-ReLU2a	ReLU	IE-Conv2a	-	-	$56 \times 56 \times 128$
IE-Conv2b	2D Conv	IE-ReLU2a	3×3	1×1	$56 \times 56 \times 128$
IE-ReLU2b	ReLU	IE-Conv2b	-	-	$56 \times 56 \times 128$
IE-MaxPool2	2D Max Pool	IE-ReLU2b	2×2	2×2	$28 \times 28 \times 128$
IE-Conv3a	2D Conv	IE-MaxPool2	3×3	1×1	$28 \times 28 \times 256$
IE-ReLU3a	ReLU	IE-Conv3a	-	-	$28 \times 28 \times 256$
IE-Conv3b	2D Conv	IE-ReLU3a	3×3	1×1	$28 \times 28 \times 256$
IE-ReLU3b	ReLU	IE-Conv3b	-	-	$28 \times 28 \times 256$
IE-Conv3c	2D Conv	IE-ReLU3b	3×3	1×1	$28 \times 28 \times 256$
IE-ReLU3c	ReLU	IE-Conv3c	-	-	$28 \times 28 \times 256$
IE-MaxPool3	2D Max Pool	IE-ReLU3c	2×2	2×2	$14 \times 14 \times 256$
IE-Conv4a	2D Conv	IE-ReLU2b	3×3	1×1	$14 \times 14 \times 128$
IE-ReLU4a	ReLU	IE-Conv4a	-	-	$14 \times 14 \times 128$
IE-Conv4b	2D Conv	IE-ReLU3c	3×3	1×1	$28 \times 28 \times 128$
IE-ReLU4b	ReLU	IE-Conv4b	-	-	$28 \times 28 \times 128$
IE-Conv4c	2D Conv	IE-MaxPool3	3×3	1×1	$56 \times 56 \times 128$
IE-ReLU4c	ReLU	IE-Conv4c	-	-	$56 \times 56 \times 128$

Table 2: Network details for the content encoder E_a , which was based upon [9]. *IE* refers to ImageEncoder. The above table contains an enumeration of all layers and operations used to encode the input frame x^0 into an appearance feature map. Note that the input column contains either the tensor used as input to the particular layer or the layer whose output is used as input. Since the proposed method involves hierarchical transformation, there are three outputs from this network: ImageEnc-ReLU4a, ImageEnc-ReLU4b, and ImageEnc-ReLU4c. The bottom section of the table indicates three extra convolutional layers, one for each output, used to force a uniform number of channels for all three encodings.

Name	Layer	Input	Kernel Dims	Strides	Output Dims
DiscV-Conv1 (skip)	3D Conv	v / \hat{v}	$4 \times 4 \times 4$	$2 \times 2 \times 2$	$8 \times 56 \times 56 \times 64$
DiscV-Conv2 (skip)	3D Conv	DiscV-Conv1	$3 \times 3 \times 3$	$1 \times 1 \times 1$	$8 \times 56 \times 56 \times 32$
DiscV-Conv3 (skip)	3D Conv	DiscV-Conv2	$3 \times 3 \times 3$	$1 \times 1 \times 1$	$8 \times 56 \times 56 \times 16$
DiscV-Conv4 (skip)	3D Conv	DiscV-Conv3	$3 \times 3 \times 3$	$1 \times 1 \times 1$	$8 \times 56 \times 56 \times 8$
DiscV-Conv5 (skip)	3D Conv	DiscV-Conv4	$4 \times 4 \times 4$	$2 \times 2 \times 2$	$4 \times 56 \times 56 \times 3$

Table 3: Network details for the Video discriminator, which takes in the generated video v and actual video \hat{v} .

4 Additional Results

Hierarchical motion integration We observe that the hierarchical motion module has significant impact on the consistent motion generation of a video. We show the qualitative difference of using a generative model without any motion module (**BaseNet-1**) and a generative model with a hierarchical motion module (**LARNet-MI-3**) in Figure 5. In Figure 5, we show how the motion module affects consistent motion generation for the NTU-RGB+D dataset [9]. For each action, the results from **BaseNet-1** often blur out and lose the motion information after few frames. In contrast, the results from **LARNet-MI-3** with motion

Name	Layer	Input	Kernel Dims ($T \times H \times W$)	Strides ($T \times H \times W$)	Output Dims ($T \times H \times W \times C$)
DiscA-Conv1	3D Conv	e_m / \hat{e}_m	$3 \times 3 \times 3$	$1 \times 1 \times 1$	$4 \times 14 \times 14 \times 64$
DiscA-Conv2	3D Conv	DiscA-Conv1	$3 \times 3 \times 3$	$1 \times 1 \times 1$	$4 \times 14 \times 14 \times 32$
DiscA-Conv3	3D Conv	DiscA-Conv2	$3 \times 3 \times 3$	$1 \times 1 \times 1$	$4 \times 14 \times 14 \times 16$
DiscA-Conv4	3D Conv	DiscA-Conv3	$3 \times 3 \times 3$	$1 \times 1 \times 1$	$4 \times 14 \times 14 \times 8$
DiscA-Conv5	3D Conv	DiscA-Conv4	$3 \times 3 \times 3$	$1 \times 1 \times 1$	$4 \times 14 \times 14 \times 8$

Table 4: Network details for the motion discriminator D_m , which takes generated representation e_m and motion representation \hat{e}_m which is extracted from a real video.

Name	Layer	Input	Kernel Dims ($T \times H \times W$)	Strides ($T \times H \times W$)	Output Dims ($T \times H \times W \times C$)
ActGen-Conv1	2D Conv	$z + p_e + a_e$	4×4	2×2	$2 \times 2 \times 128$
ActGen-Conv2	2D Conv	ActGen-Conv1	4×4	2×2	$4 \times 4 \times 128$
ActGen-Conv3	3D Conv	ActGen-Conv2	$4 \times 4 \times 4$	$2 \times 2 \times 2$	$2 \times 8 \times 8 \times 128$
ActGen-Conv4	3D Conv	ActGen-Conv3	$4 \times 4 \times 4$	$2 \times 2 \times 2$	$4 \times 16 \times 16 \times 128$
ActGen-Conv5	3D Conv	ActGen-Conv4	$1 \times 3 \times 3$	$1 \times 1 \times 1$	$4 \times 14 \times 14 \times 128$
ActGen-Conv6	3D Conv	ActGen-Conv5	$4 \times 4 \times 4$	$2 \times 2 \times 2$	$4 \times 28 \times 28 \times 128$

Table 5: Network details for motion generator G_m , which takes in stochastic noise z , position encoding p_e and word embedding a_e and generates a latent action representation e_m

Name	Layer	Input	Kernel Dims ($T \times H \times W$)	Strides ($T \times H \times W$)	Output Dims ($T \times H \times W \times C$)
AppT-Conv1	2D Conv	ActT-Conv3(1) + KPD-Gaussian + ImageEnc-ReLU4a	7×7	1×1	$14 \times 14 \times 256$
AppT-Split	Split	AppT-Conv1	-	-	$14 \times 14 \times 128$
AppT-Sig1	Sigmoid	AppT-Split(1)	-	-	$14 \times 14 \times 128$
AppT-Sig2	Sigmoid	AppT-Split(2)	-	-	$14 \times 14 \times 128$
AppT-Conv2	2D Conv	ActT-Conv3(1) + KPD-Gaussian + AppT-Sig1	7×7	1×1	$14 \times 14 \times 128$
AppT-Tanh	Tanh	ImageEnc-ReLU4a AppT-Conv2	-	-	$14 \times 14 \times 128$
AppT-Final	Concat	(1 - AppT-Sig2) * ImageEnc-ReLU4a + AppT-Sig2 * AppT-Tanh	-	-	$14 \times 14 \times 128$

Table 6: Network details for the Motion Integrator, M_I , which is used to transform the appearance embeddings from the content encoder E_a according to the transformed action features and the predicted action key-points. Note that t_A is a recurrent network and only one of the recurrent cells is detailed above. This cell would be repeated T times, where T is the size of the temporal dimension of the transformed action features. Then, the T cell outputs are concatenated in the temporal dimension to produce a transformed appearance of the same size as the transformed action features. Also, note that hierarchical transformation is used, so the recurrent network is used three times, once for each of the appearance embeddings.

module shows how the model generates motion for the corresponding action.

Name	Layer	Input	Kernel Dims ($T \times H \times W$)	Strides ($T \times H \times W$)	Output Dims ($T \times H \times W \times C$)
VidGen-Conv1a	3D Conv	AppT1+ActEnc1	$4 \times 4 \times 4$	$2 \times 2 \times 2$	$8 \times 28 \times 28 \times 256$
VidGen-Conv1b	3D Conv	VidGen-Conv1a	$3 \times 3 \times 3$	$1 \times 1 \times 1$	$8 \times 28 \times 28 \times 256$
VidGen-Conv2a	3D Conv	AppT2 + VidGen-Conv1b	$4 \times 4 \times 4$	$2 \times 2 \times 2$	$16 \times 56 \times 56 \times 256$
VidGen-Conv2b	3D Conv	VidGen-ReLU2a	$3 \times 3 \times 3$	$1 \times 1 \times 1$	$16 \times 56 \times 56 \times 128$
VidGen-Conv3a	3D Conv	AppT3 + VidGen-Conv2b	$1 \times 4 \times 4$	$1 \times 2 \times 2$	$16 \times 112 \times 112 \times 128$
VidGen-Conv3b	3D Conv	VidGen-Conv3a	$3 \times 3 \times 3$	$1 \times 1 \times 1$	$16 \times 112 \times 112 \times 64$
VidGen-Conv4a	3D Conv	VidGen-Conv3b	$3 \times 3 \times 3$	$1 \times 1 \times 1$	$16 \times 112 \times 112 \times 8$
VidGen-Conv4b	3D Conv	VidGen-Conv4a	$1 \times 1 \times 1$	$1 \times 1 \times 1$	$16 \times 112 \times 112 \times 3$
VidGen-Sig	Sigmoid	VidGen-Conv4b	-	-	$16 \times 112 \times 112 \times 3$

Table 7: Network details for the video decoder, G_v , which generates the final output video v based on the latent video features e_v generated using motion integration M_I .

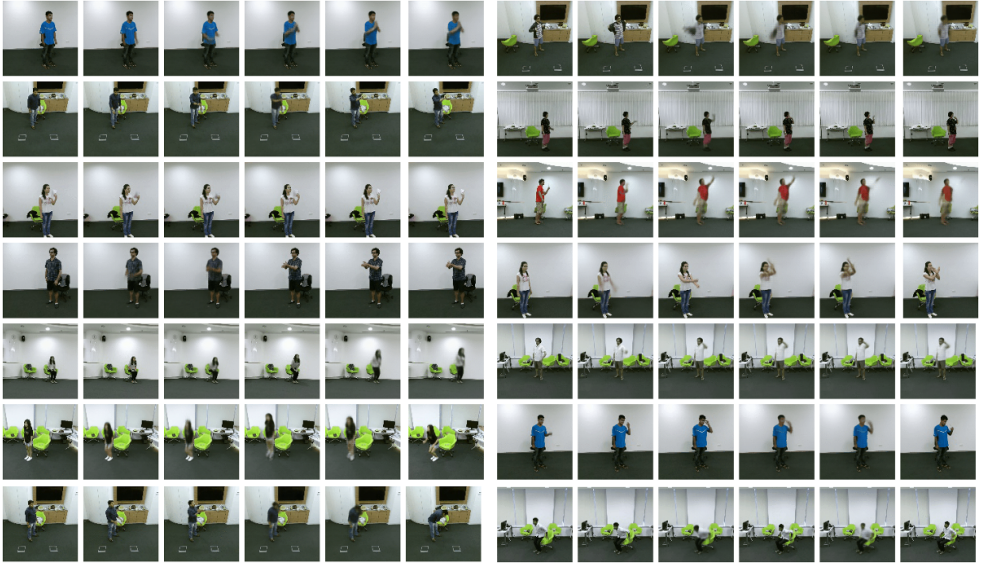


Figure 1: Generated videos using our method on the NTU dataset. Row 1: *rub two hands, take off jacket*, Row 2: *check time from watch, phone call*, Row 3: *fan self, throw*, Row 4: *point finger, throw*, Row 5: *stand up, salute*, Row 6: *sit down, take off glasses*, Row 7: *sneeze cough, sit down*.

Comparison with previous methods We show the qualitative comparison against conditional VGAN [10], G³AN [11], MoCoGAN [12] and ImaGINator [13] for the NTU-RGB+D dataset [8] in Figure 3. VGAN [10] uses first frame as conditional input and uses a separate foreground and background stream, which makes it favorable to produce videos with static

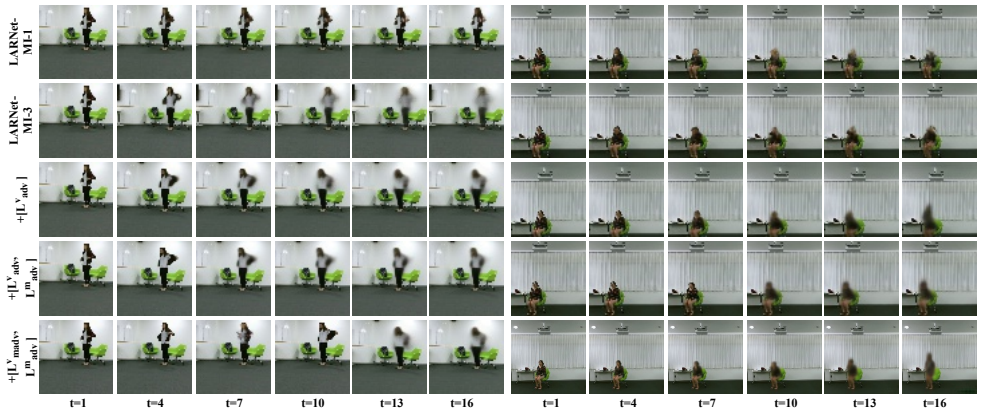


Figure 2: Demonstration of effects of various components of the proposed LARNet for video generation. It can be observed that with low motion integration the video tries to repeat first frame in row 1. As we add motion and various losses, the generated video quality improves (row 2 - 5). Row 1: $LARNet - MI - 1$, Row 2: $LARNet - MI - 3$, Row 3: $+ [L^v_{adv}]$, Row 4: $+ [L^v_{adv}, L^m_{adv}]$, Row 5: $+ [L^v_{adv}, L^m_{adv}]$.

images. G³AN [10] and MoCoGAN [11] use the class label as conditional input.

Action representation synthesized from one-hot vectors We use the class label as one-hot encoding instead of using action embedding a_e from GloVe-300 to generate action representation. The full network is trained using in the same manner other than the change in input action encoding vector. The quantitative scores are shown in Table 8.

No action representation We use the class label as one-hot encoding instead of using action embedding a_e from GloVe-300 to generate action representation. This variation uses skip connections from appearance encoder without requiring any motion integration module. The full network is trained using a reconstruction MSE loss and an adversarial loss on the generated video. The quantitative scores are shown in Table 8.

Approach	FID ↓	FVD ↓	PSNR ↑	SSIM ↑
$LARNet_{1-hot}$	177.02	14.76	25.64	0.92
$LARNet_{1-hot, MI-3}$	168.34	14.09	27.11	0.93
$LARNet_{GloVe, MI-3}$	165.53	13.34	28.4	0.94

Table 8: Comparison of using one-hot encoding and GloVe-300 word encoding for the labels in our LARNet model. $LARNet_{1-hot}$ uses only one-hot encoding for the labels and has no hierarchical motion integrator. $LARNet_{1-hot, MI-3}$ uses one-hot encoding for the labels and has hierarchical motion integrator (MI - 3). $LARNet_{GloVe, MI-3}$ uses the GloVe-300 text encoding for labels and has hierarchical motion integrator (MI - 3).

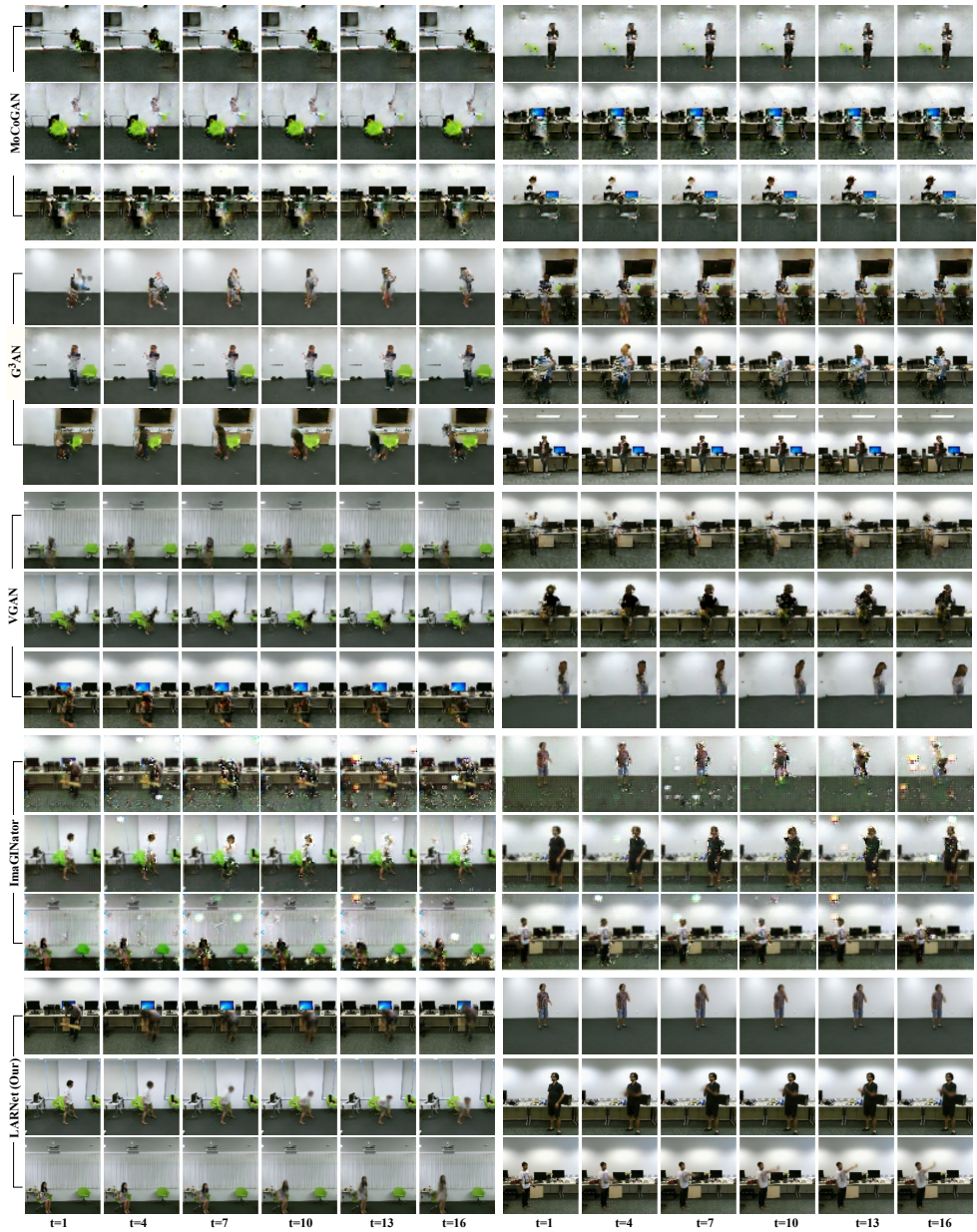


Figure 3: Comparison with previous conditional state-of-the-art methods on the NTU-RGB+D dataset.

Evaluation on foreground only To further compare the effects of the various components in our proposed method, we evaluate the metrics for the foreground region only. The NTU-RGB+D dataset provides the ground truth skeletal data which can be used for cropping out a region around the actor. Since our videos have large static background, it becomes harder to

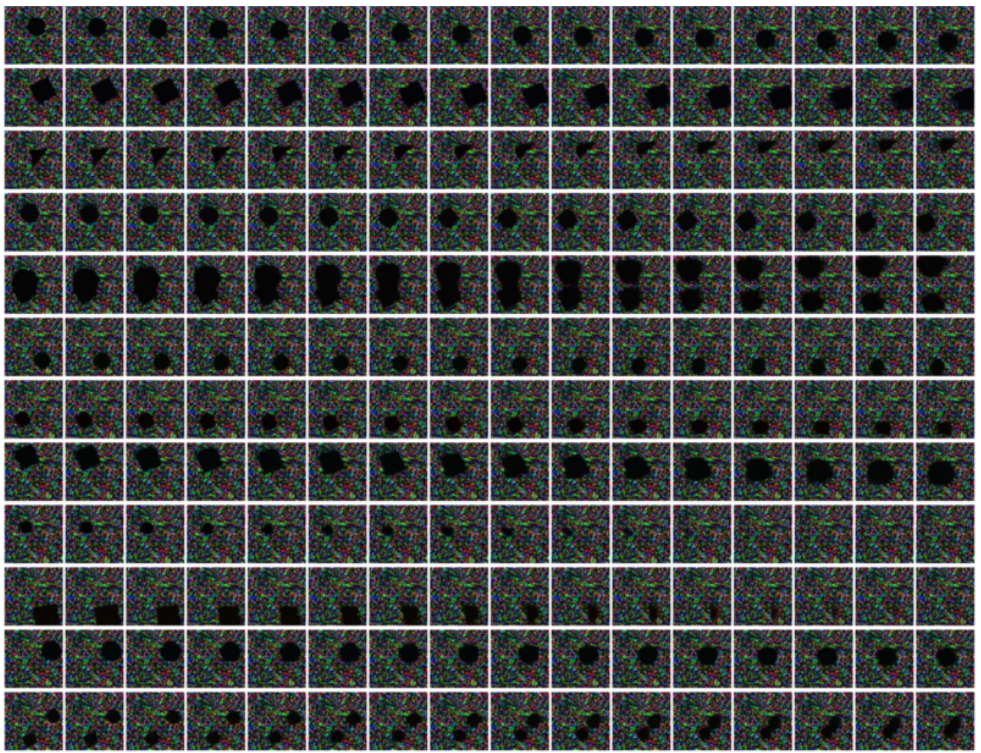


Figure 4: Generating videos using our method on the Synthetic dataset for all 6 synthetic classes *move right to left*, *move left to right*, *move down to up*, *move up to down*, *split* and *move apart*, *merge together*.

evaluate the difference between different network components as most of the region seems realistic. To remedy this, we crop the foreground region and compute all scores on it. The score is reported in Table 9. The computations for various ablation is done in a lower spatial dimension.

Demonstration of various network components We visualize the effect of various network components used in LARNet full model in Figure 2. We show how the generated video differs with addition on each component in the network. First we show video generation with only one motion integrator denoted by $LARNET - MI - 1$, where the video tries to imitate first frame and has very little motion. This makes the video look sharp but it has no movement. Then we show the effect of using three motion integrators denoted by $LARNET - MI - 3$, where the video has blurry motion. We then show how adding each loss term improves the generated video, with the final model having $LARNet + [L_{adv}^v, L_{adv}^m]$ and giving the best generated video.

Qualitative Results We provide more qualitative results from our method on the UTD-MHAD [2] dataset in Figure 6.

We provide more qualitative results from our method on the Synthetic dataset in Figure 4.



Figure 5: Ablation to study the effect of hierarchical motion integrator (MI) on the base version of LARNet. We show the difference between using only the **Basenet-1** and adding the hierarchical motion integrator **MI** with the **Basenet-1 (NOT FULL NETWORK)** on NTU-RGB+D dataset. Row 1: *stand up*, Row 2: *salute*, Row 3: *stand up*, Row 4: *stand up*, Row 5: *hand waving*, Row 6: *put on a hat cap*.

Approach	FID ↓	FVD ↓	PSNR ↑	SSIM ↑
$LARNet_{MI-3}$	144.67	9.67	31.84	0.900
$+L_{adv}^m$	145.29	9.79	30.86	0.923
$+L_{adv}^v$	143.57	8.66	33.45	0.945

Table 9: Ablation scores for NTU-RGB+D dataset on foreground region only.

We provide more qualitative results from our method on the NTU-RGB+D dataset in Figure 1.

References

- [1] Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *CVPR*, 2017.
- [2] Chen Chen, Roozbeh Jafari, and Nasser Kehtarnavaz. Utd-mhad: A multimodal dataset for human action recognition utilizing a depth camera and a wearable inertial sensor. In *2015 IEEE International conference on image processing (ICIP)*, pages 168–172. IEEE, 2015.
- [3] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash

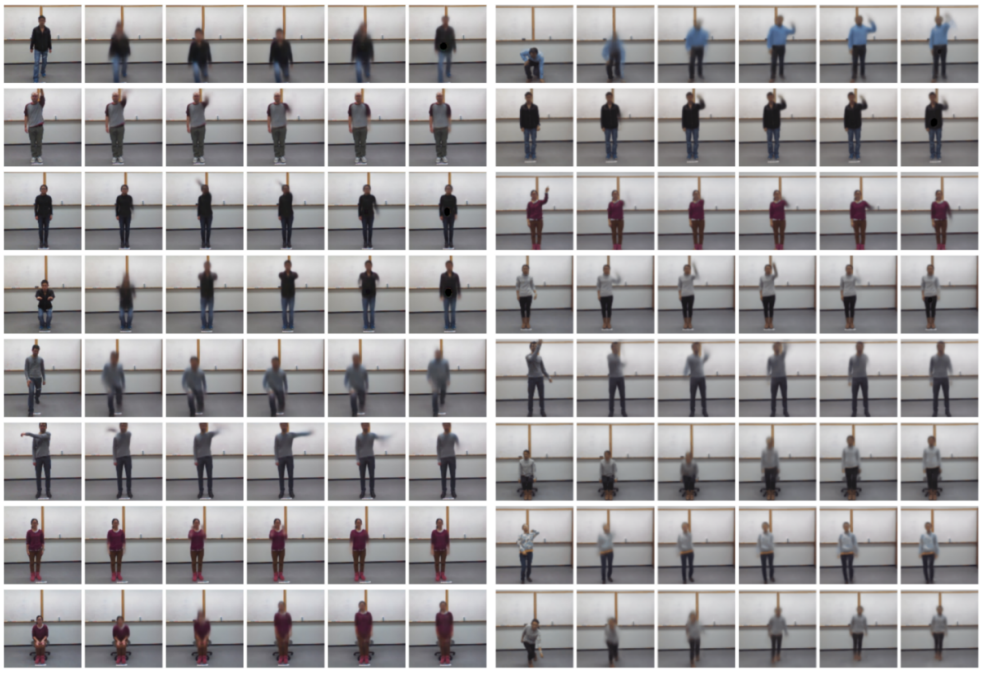


Figure 6: Generated videos using our method on the UTD-MHAD dataset. Row 1: *lunge, sit to stand*, Row 2: *tennis swing, catch*, Row 3: *wave, wave*, Row 4: *sit to stand, wave*, Row 5: *lunge, tennis serve*, Row 6: *swipe right, sit to stand*, Row 7: *arm cross, tennis serve*, Row 8: *sit to stand, bowling*.

equilibrium. In *Advances in neural information processing systems*, pages 6626–6637, 2017.

- [4] Alain Hore and Djemel Ziou. Image quality metrics: Psnr vs. ssim. In *2010 20th International Conference on Pattern Recognition*, pages 2366–2369. IEEE, 2010.
- [5] Amir Shahroudy, Jun Liu, Tian-Tsong Ng, and Gang Wang. Ntu rgb+ d: A large scale dataset for 3d human activity analysis. In *Proceedings of the IEEE conference on CVPR*, 2016.
- [6] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.
- [7] Sergey Tulyakov, Ming-Yu Liu, Xiaodong Yang, and Jan Kautz. Mocogan: Decomposing motion and content for video generation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.
- [8] Thomas Unterthiner, Sjoerd van Steenkiste, Karol Kurach, Raphael Marinier, Marcin Michalski, and Sylvain Gelly. Towards accurate generative models of video: A new metric & challenges. *arXiv preprint arXiv:1812.01717*, 2018.

- [9] Thomas Unterthiner, Sjoerd van Steenkiste, Karol Kurach, Raphaël Marinier, Marcin Michalski, and Sylvain Gelly. Fvd: A new metric for video generation. 2019.
- [10] Carl Vondrick, Hamed Pirsiavash, and Antonio Torralba. Generating videos with scene dynamics. In *NeurIPS*, 2016.
- [11] Yaohui WANG, Piotr Bilinski, Francois Bremond, and Antitza Dantcheva. Imaginator: Conditional spatio-temporal gan for video generation. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, March 2020.
- [12] Yaohui Wang, Piotr Bilinski, Francois Bremond, and Antitza Dantcheva. G3an: Disentangling appearance and motion for video generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5264–5273, 2020.
- [13] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004.